НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і
спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____В.П. Тарасенко
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050102 «Комп'ютерна інженерія»

на тему: «Модуль керування автоматизованої зарядної станції електромобіля»

Виконав (-ла):
студент (-ка) IV курсу, групи КВ-52
Затилюк Дмитро Олександрович _____

Керівник:

_____

Консультант з нормоконтролю:

_____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____В. П. Тарасенко

«___»_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Затилюку Дмитру Олександровичу

1. Тема проекту «Модуль керування автоматизованої зарядної станції електромобіля», керівник проекту Коляда Костянтин В'ячеславович, асистент каф. СПСКС, затверджені наказом по університету від 22.05.2019 р. №1330-С

2. Термін подання студентом роботи « » червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст роботи:
   - аналіз існуючих рішень, обґрунтування теми диплому;
   - апаратна розробка;
   - розробка серверної частини.

5. Перелік обов'язкового графічного матеріалу:
   - структурна схема;
   - функціональна схема;
   - схема роботи додатку;
   - принципова схема пристрою.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль			

7. Дата видачі завдання «__» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за тематикою роботи та збір даних	01.04.2019	
2.	Розроблення технічного завдання	08.04.2019	
3.	Розроблення структури проекту	19.04.2019	
4.	Розроблення функціоналу обробки запитів на створення таблиць	28.04.2019	
5.	Підготовка матеріалів першого розділу дипломного проекту	03.05.2019	
6.	Розробка апаратної частини проекту	14.05.2019	
7.	Підготовка матеріалів другого розділу дипломної роботи	21.05.2019	
8.	Підготовка матеріалів третього розділу дипломної роботи	25.05.2019	
9.	Підготовка графічної частини дипломної роботи	30.05.2019	
10.	Оформлення дипломної роботи	02.06.2019	

Студент Д.О. Затилюк

Керівник роботи К.В. Коляда

АНОТАЦІЯ

Дана дипломна робота присвячена розробці модуля керування автоматизованої зарядної станції електромобіля.

В роботі проведено аналіз методів побудови існуючих зарядних пристроїв електромобіля, та враховуючи всі недоліки сформульовані вимоги до модуля керування автоматизованої зарядної станції.

Також розроблено програмне забезпечення контролера для управління процесом зарядки автомобіля та розроблено користувацький додаток для управління та моніторингу роботи зарядної станції , що дасть повний контроль користувачу над процесом.

ABSTRACT

This thesis is devoted to the development of the control module of the automated charging station of the electric vehicle.

The paper analyzes the methods of constructing existing charging devices for electric vehicles, and taking into account all the disadvantages formulated requirements for the control module of the Automated Charging Station.

A controller software for controlling the process of charging a car has also been developed and is developed by the advisory application for controlling and monitoring the work of the charging station, which will give full control of the user over the process.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.468300.002 ТЗ	Модуль керування	4		
			автоматизованої зарядної			
			станції електромобіля			
			Технічне завдання			
	А4	ІАЛЦ.468300.003 ТП	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.468300.004 ПЗ	Модуль керування	52		
			автоматизованої зарядної			
			станції електромобіля			
			Пояснювальна записка			
	А4	ІАЛЦ.468300.005 Д1	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Схема підключення			
			системи			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ.468300.001 ОА		
Розроб.		Затилюк Д.О.			Модуль керування автоматизованої зарядної станції електромобіля *Опис альбому*	Літ.	Лист 1 / Листів 2
Перев.		Коляда К.В.					
Н. контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-52	
Затв.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.468300.006 Д2	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Система алгоритму роботи			
			апаратної частини. Схема			
			структурна.			
	А4	ІАЛЦ.468300.007 Д3	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Взаємодія модулів			
			програми. Структура.			
	А4	ІАЛЦ.467200.008 Д4	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Схема алгоритму роботи			
			серверної частини.			
			Схема структурна.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.468300.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Модуль керування автоматизованої зарядної станції електромобіля».

Галузь застосування: електричний транспорт.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи ступеня «бакалавр комп'ютерної інженерії», затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення смостійного пристрою зарядної станції з керуванням через мобільний додаток без використання людської праці.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою (IOS,Android,WP);
- можливість управління процесом зарядки;
- можливість поповнення рахунку клієнта;

5.2. Вимоги до апаратного забезпечення

- Мікропроцесор: 304 MIPS ;
- Пам’ять:
    - 64 MB SDRAM
    - 32 MB Flash chip
    - 512 Kb of Non-volatile SRAM;
- Наявність доступу до мережі Internet (Ethernet, Wi-Fi);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows Phone, Android, iOS;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.468300.004 ПЗ	Модуль керування	52		
			автоматизованої зарядної			
			станції електромобіля			
			Пояснювальна записка			
	А4	ІАЛЦ.468300.005 Д1	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Схема підключення			
			приладів			
	А4	ІАЛЦ.468300.006 Д2	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Схема алгоритму роботи			
			апаратної частини.			
			Схема структурна			
	А4	ІАЛЦ.468300.007 Д3	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Взаємодія модулів			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.468300.003 ТП		
Розробив		Затилюк Д.О.			Модуль керування автоматизованої зарядної станції електромобіля Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Коляда К.В.					
Консульт.						**КПІ ім. Ігоря Сікорського, ФПМ, КВ-52**	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			програми. Структура.			
	А4	ІАЛЦ.468300.008 Д4	Модуль керування	1		
			автоматизованої зарядної			
			станції електромобіля			
			Схема алгоритму роботи			
			серверної частини.			
			Схема структурна.			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

STRUM – це швидкі суперчарджери виробництва відомої шведсько-швейцарської компанії ABB, що спеціалізується в області електротехніки, енергетичного машинобудування та інформаційних технологій.

Суперчарджер – зарядна електростанція, що має високу ефективність та малу швидкість зарядки акумулятора автомобіля. Назва пішла від Tesla Supercharger – перша швидка зарядка електромобіля від компанії Tesla.

Tesla Inc — американська автомобільна компанія-стартап із Кремнієвої долини. Орієнтована на дизайн, виготовлення та продаж електромобілів та компонентів до них.

«Groupe Renault» — транснаціональна група підприємств і компаній, яка виробляє кілька марок автомобілів та має промислові об'єкти і комерційні установи в 134 країнах Світу.

ЕЗС – електрозаправна станція.

ПЛК – програмований логічний контролер.

ROM, ПЗП – постійний запам’ятовуючий пристрій.

RAM, ОЗП – оперативна пам’ять.

ЦП – центральний процесор – функціональна частина комп'ютера, що призначена для інтерпретації команд.

PIC — сукупність сімейств 8-ми, 16-ти розрядних та 32-х розрядних мікроконтролерів, що мають гарвардську архітектуру. Випускаються фірмою Microchip.

AVR — родина восьмибітових мікроконтролерів фірми Atmel. Мікроконтролери AVR мають гарвардську архітектуру і систему команд, близьку до ідеології RISC.

Forth – одна з перших конкатенативних мов програмування, в якій програми записуються послідовністю лексем.

Мова асемблера – машинно-орієнтована мова програмування низького рівня.

Мова С – компільована статично типізована мова програмування загального призначення, розроблена в 1969-1973 роках співробітником Bell Labs Деннісом Рітчі як розвиток мови Бі.

Бейсік – сімейство високорівневих мов програмування. Стала популярною насамперед як мова для домашніх комп'ютерів.

Паскаль – одна з найбільш відомих мов програмування, використовується для навчання програмуванню в старших класах і на перших курсах вузів, є основою для ряду інших мов.

С++ – компільована, статично типізована мова програмування загального призначення.

ІЧ – інфрачервоний сигнал передачі даних.

ООП (Об'єктно-орієнтоване програмування) – методологія програмування, що основана на представлення програми у вигляді сукупності об'єктів, кожен з якого являється екземпляром певного класу, а класи складають ієрархію наслідування.

RS485 – стандарт фізичного рівня для асинхронного інтерфейсу. Регламентує електричні параметри напівдуплексної багато точкової диференціальної лінії зв'язку типу «загальна шина».

HTML (HyperText Markup Language) - стандартний язык розмітки гіпертекстових сторінок в Інтернеті. Є і інші мови розмітки гіпертекста, але велика частина сторінок Інтернету розміщується саме на мові HTML. Такі сторінки успішно інтерпретують браузери, які відображають їх на екранах різних електронних пристроїв в зручному для людини вигляді.

CSS (англ. Cascading Style Sheets, каскадні таблиці стилів) - це проста мова дизайну, призначена для втілення процесу презентації веб-сторінки.

JavaScript - це спеціальний мовний програмування, розроблений для використання в браузерах. Використовується JavaScript для написання

невеликих програм - скриптів, які можуть реагувати на цільові дії користувачів, відкриваючи ту чи іншу веб-сторінку. Хороша ілюстрація використання JavaScript буде випускати меню, які використовують для навігації на багатьох сайтах у Всесвітній павутині.

REST (Representational state transfer) - це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови веб-служб.

AJAX - це абревіатура, яка означає Asynchronous Javascript and XML. При використанні AJAX немає необхідності оновлювати кожен раз всю сторінку, так як оновлюється тільки її конкретна частина. Це набагато зручніше, так як не доводиться довго чекати, і економічніше, так як не всі мають безлімітний та швидкий інтернет.

ВСТУП

В умовах сучасного світу все актуальнішою проблемою являється чисте довкілля. Саме тому більшість автомобільних компаній розробляють нові моделі електромобілів. Лиш за останні два роки кількість екологічних авто збільшилась щонайменше вдвічі. Тому, для комфортного користування електрокарами, потрібно забезпечити автошляхи достатньою кількістю зарядних станцій.

Мета виконання дипломної роботи полягає у створенні зручного модуля для зарядної станції, що дасть змогу користувачу повністю контролювати зарядний процес.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Дослідження в області інфраструктури електромобілів.

Група експертів Офісу ефективного регулювання BRDO зробила висновок, що одним з основних перешкод переходу на масове використання електромобілів в Україні є недостатній рівень розвитку інфраструктури електрозарядних станцій, яка, в свою чергу, не розвивається через відсутність законодавчої бази, стимулів і інвестицій з боку держави і приватного бізнесу.

Для зростання кількості електричних автомобілів потенційні власники електромобілів повинні бути впевнені в тому, що можуть отримати доступ до розвиненої і зручною для них мережі зарядних станцій. При цьому, інформація про розташування електрозарядних станцій і їх ціновій політиці повинна бути загальнодоступною.

Кожен користувач повинен мати можливість оплачувати послуги в зручний для нього спосіб. Система зберігання та передачі інформації, проведення транзакцій повинна бути зручною і безпечною для споживача.

Держава та бізнес повинні робити все можливе, щоб кожне 10-е місце для паркування на великих парковках і автостоянках буде обладнано ЕЗС, купувати електроенергію в онлайн режимі у будь-якого постачальника, віддавати електроенергію в мережу в онлайн режимі і отримувати за це плату.

Наша дипломна робота вносить свій вклад в розвиток цієї інфраструктури. Оскільки теперішні реалії як в країні, так і в світі не такі позитивні як хотілося б.

Згідно досліджень світових експертів, в країнах, де автомобілів на електротязі вже досить багато, близько 90% заправки електроенергією припадає на власне житло власника такого автомобіля. І лише 10% - це

електрозаправки, що знаходяться в громадських місцях чи на дорогах. Але саме вони дають гарантію будь якому власнику електроавтомобіля відчуття спокою та впевненості, що його транспорт не залишиться нерухомим посеред дороги без підзарядки. Тому, потрібно розширяти кількість електрозаправних станцій, щоб вони зустрічались на дорозі мінімум кожні 50-100 кілометрів.

1.2. Аналіз ринку ЕЗС.

Нині на території України рішеннями для зарядки електромобілів являються такі аналоги:

- суперчарджери STRUM
- електрозарядні станції на заправках та парковках
- домашні зарядні станції

Також, кожна компанія, що випускає електромобілі, намагається забезпечити своїх клієнтів достатньою кількістю зарядних станцій. В такому напрямку розвиваються такі компанії як Tesla, Renault та інші.

Електрозарядні станції компанії STRUM – це так звані суперчарджери. Тобто ці станції дуже швидко заряджають автомобіль. Зазвичай, сеанс триває від 20 хвилин до однієї години, в залежності від ємності акумулятора.

Так, ці зарядні станції є досить потужними та швидкими, але вона має і свої недоліки. Головна проблема, що породжує й інші це те, що така швидко зарядна станція потребує досить потужну мережу електроенергії. В свою чергу це обумовлює таку малу кількість таких електростанцій на території країни.

Наступними аналогами є станції на заправках та парковках. Зазвичай це примітивні та повільні зарядні станції, що не мають ніякого інтерфейсу керування для користувача. Для оплати та керування зарядним процесом зазвичай використовується людський фактор – оператор проводить увімкнення розетки з централізованого пульту, або ж ціна зарядки входить в оплату перебування на парковці.

Останнім аналогом являються домашні зарядні станції. Для цього купується спеціальний шнур який має вихід на звичайну розетку. Тому, електромобіль можна зарядити прямо в гаражі. Найбільшим мінусом для

такого способу є те, що середній час зарядки займе від 12 годин. Також це може перенавантажити мережу.

Також, дослідження компанії МОЭСК–EV показали, що хоч і кількість електромобілів стрімко росте, але попит на спеціалізовані електростанції не так швидко збільшується, а такі станції є не найголовнішою проблемою користувачів екологічних автівок.

Майже 90% зарядного часу зарядної сесії автівок проходить якраз таки в домашніх умовах. Це малоефективно та не досить позитивно відображається на домашній електромережі. Але користувачам більш зручно заряджати свої автівки ближче до свого дому.

1.3. Обгрунтування теми дипломного проекту.

З наведених вище зібраних даних ми розуміємо, що електромобільній інфраструктурі ще є куди і дуже потрібно розвиватися.

Наш пристрій може позитивно посприяти темпам розвитку цієї області. Тому, що автоматизований прилад зменшує затрати людської праці, дає можливість збільшувати кількість станцій та спрощує процес управління сеансом.

Завданням нашого дипломного проекту буде створення такого модуля, який можна буде вбудувати в будь яку станцію, як суперчарджер так і звичайна зарядна станція тих чи інших виробників.

Такі станції повинні працювати без оператора і бути підключенні до однієї системи та бази даних. Інтерфейс управління та функціонал повинен бути досить зрозумілим, щоб користувач міг повністю контролювати зарядний процес, а також повинен мати можливість поповнення свого віртуального рахунку.

В першу чергу ці всі функції повинні бути реалізована за для того, щоб такі станції можна було поставити в будь яке місце де проходить електромережа. Тобто, як на парковки, в двори багатоповерхівок, так і по серед дороги в будь якій точки країни, як наприклад якесь невеличке село чи гори. Як повздовж магістралі, так і на невеличких дорогах районного типу.

2. ПРОЕКТУВАННЯ АПАРАТНОЇ ЧАСТИНИ

2.1. Проектування системи.

Першим етапом розробки апаратної частини являється проектування системи. Задача полягає в тому, щоб скласти список компонентів та спроектувати взаємодію їх між собою.

Основним пристроєм системи, так названим мозком нашого модуля, має бути якийсь центральний комп’ютер управління пристроями. Його функцію можуть виконувати як мікроконтролер або ж ПЛК, так і інтегральний контролер. Контролер буде отримувати дані з серверу та на їх підставі вмикати ту чи іншу розетку для підзарядки автомобіля.

Наступним елементом є реле. Реле виконує досить просту функцію в нашому приладі – функція вимикача. Тобто, наше реле подаватиме струм або ж навпаки вимикатиме подачу в залежності від стану оплати за кількість струму або ж при завершенні зарядки.

Також, в нашій системі буде присутній лічильник електроенергії. Його функція досить проста – підрахувати скільки ми вже передали кВт та передавати це значення на контролер.

В нашому випадку краще буде використовувати підвид реле – контактори. Їх перевага перед стандартними реле в тому, що вони призначені для частих дистанційних вмикань та вимикань силових електричних кіл.

2.2. Вибір компонентів апаратної частини модуля.

Першим, можна сказати, головним елементом всієї апаратної частини є наш прилад управління пристроями – так званий мікрокомп’ютер. Він являється локальним мозком приладу. Як ми вже визначали, таким пристроєм може являтися як мікроконтролер, ПЛК, інтегральний контролер тощо.

Отже, спершу пропозиція була використати якийсь популярний мікроконтролер.

Мікроконтролер - це спеціальна мікросхема, призначена для управління різними електронними пристроями. Розробники мікроконтролерів вигадали дещо нове на той час - об'єднати процесор, пам'ять, ПЗП і периферію всередині одного корпусу, зовні схожого на звичайну мікросхему. З тих пір виробництво мікроконтролерів щорічно у багато разів перевищує виробництво процесорів, а потреба в них не знижується.

Мікроконтролери випускають десятки компаній, причому виробляються не тільки сучасні 32-бітові мікроконтролери, а й 16, і навіть 8-бітові. Усередині кожної родини часто можна зустріти майже однакові моделі, що розрізняються швидкістю роботи ЦП та обсягом пам'яті.

Популярністю у розробників користуються 8-бітові мікроконтролери PIC фірми Microchip Technology і AVR фірми Atmel, 16-бітові MSP430 фірми TI, а також 32-бітові мікроконтролери, архітектури ARM, яку розробляє фірма ARM Limited і продає ліцензії іншим фірмам для їх виробництва.

Не зважаючи на те, що область застосування мікроконтролерів переважно у вбудованих системах. Тобто, їх використовують в іграшках, в верстатах, в масовій побутовій техніці, та автоматизації деяких систем будівель. На жаль, їх основний плюс і є причиною великого мінуса. З причини своїх мініатюрних розмірів, що звичайно для нас досить важливо і зручно, вони мають досить обмежені характеристики. Зазвичай вони

використовуються там, де потрібна не потужність процесора, а, скоріше, баланс між ціною і достатньою функціональністю.

Звичайно, результатом складання з декількох мікропроцесорів однієї модульної системи збільшать обсяги пам’яті та потужності, але заодно і збільшиться складність системи та її об’єми в плані кількості компонентів.

Рис. 2.1. 16-бітний 28-pin PDIP PIC24 мікроконтролер

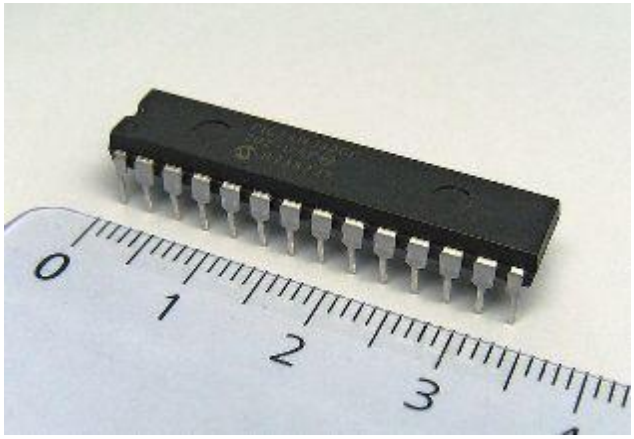


Рис. 2.2. Мікроконтролер Atmel AVR ATmega8 в корпусі DIP

Також великим мінусом являється важкість написання програми для для даних пристроїв. Оскільки в нас досить обмежені об’єми пам’яті, а зазвичай об’єми складають від 2 до 128 Кб, то потрібно продумувати все досить ретельно. Також, такі мікроконтролери зазвичай використовують досить застарілі або ж не досить зручні для сучасного програміста мови

програмування. В основному це різновиди мови програмування С. Інколи використовується асемблер, Forth або ж спеціалізовані підвиди Бейсіка чи Паскаля.

Також, серед мікроконтролерів є й більш потужні та популярні серед любителів плати такі як Arduino UNO або ж Rasberry Pi.

Рис. 2.3. Мікроконтролер Arduino UNO

Arduino UNO - найпопулярніша платформа аматорської та освітньої електроніки і робототехніки. Розглянемо її плюси і мінуси:

Плюси:

- мікроконтролер має достатньо малі розміри;
- програмування виконується на потужній мові подібній до С++;
- має досить багато бібліотек та готових рішень;
- низький поріг для новачків розробників.

Мінуси:

- дуже поганий редактор коду;

- щоб закінчити проект із застосуванням Arduino, вам доведеться вручну прошити завантажувач в кожен новий мікроконтролер ATmega. Він займає 2Кб пам'яті;
- має малі об’єми пам’яті (16 Кб флеш-пам’яті (2 Кб використовується для збереження завантажувача), 1024 байта ОЗП, 512 байт EEPROM);
- для використання досить популярного протоколу RS-485 потрібно докуповувати спеціальний модуль;
- одно поточність, що компенсується перериваннями;
- швидкість роботи досить повільна;

Отже на цьому етапі ми зробили висновок, що для такої досить важливої і самостійної розробки потрібно підібрати якийсь більш потужний, ефективний та більш безпечний продукт. Так, звичайно, в промислових цілях при виході на ринок можна буде ще раз розглянути мікроконтролери як варіант для даного модуля, але на даному етапі малими розмірами та низькою ціною можна пожертвувати в на користь більших об’ємів пам’яті, потужності мікропроцесора та простоті розробки.

Наступним кандидатом було вибрано інтегральні контролери.

Цей пристрій може бути реалізований як багатофункціональний або ж з можливістю розширення іншими модулями. Контролер може оснащуватися панеллю комутації або передавати ці функції до інших модулів.

Контролери мають підключення до мережі Інтернет, що дозволяє виконувати управління системою з мобільного пристрою, планшета, комп'ютера і можуть являти собою комп'ютери з певною кількістю портів або виготовлятися у вигляді гаджета з призначеним для користувача інтерфейсом.

У контролері може бути передбачена можливість програмування або управління комплексом, він може оснащуватися кнопками або панеллю з сенсорним управлінням. Найчастіше для програмування застосовується спеціальне програмне забезпечення або навіть окреме середовище розробки. До персонального комп'ютера контролер може підключатися за допомогою Ethernet-контролера або з використанням послідовного інтерфейсу RS-485.

Більшість сучасних інтегральних контролерів мають достатній обсяг пам’яті, потужності мікропроцесора та функціонал для реалізації нашого приладу. Тому обирати потрібно було за ціновою категорією, невеликими розмірами та звернути увагу на надійність продукту та ім’я компанії.

При проходженні практики, керівники компанії підказали мені поглянути в бік контролерів AMX. Корпорація AMX на сьогоднішній день є лідером в світі в розробці і виробництві систем управління і автоматизації приладів та нерухомості. Робота контролерів здійснюється за рахунок операційної системи реального часу VxWorks (компанія Wind River Systems, Каліфорнія, США). Що і доводить стабільність і надійність контролерів AMX.

Тому я розглянув декілька варіантів контролерів цієї компанії і звернув увагу на модель NetLinx Integrated Controller AMX NI-700.

Рис. 2.4. NetLinx Integrated Controller AMX NI-700

Дуже не дорогий, але вкрай швидкий контролер для невеликих проектів. NI-700 – пристрій, що являє собою блок управління, призначений для створення системи автоматизації та централізованого контролю обладнання в межах одного приміщення. Чудове поєднання ціни і функціональності задовольнять вимоги будь-якого замовника.

Розташовані на панелі контролера інтерфейси дозволяють об'єднати в єдину систему управління відео-плеєри, проектор, екран, освітлення, термостати, і інше електронне обладнання. Ідеально підходить для невеличких пристроїв або ж для управління системи в межах одного приміщення з обмеженою кількістю пристроїв, таких як навчальні аудиторії, конференц-зали, переговорні кімнати, готельні номери і т.д.

Також, дуже важливим фактором є те, що на переддипломній практиці ми отримали практичні та теоретичні навички програмування контролерів сімейства NetLinx Integrated Controller. Ці контролери є досить потужними та універсальними, що дозволяє використання їх при проектуванні будь яких систем автоматизацій чи створення різних приладів управління.

Розглянемо кількість портів та системні характеристики цього контролеру:

Порти:

- два RS-232 / RS-422 / RS-485 послідовних порти;
- 1 ІЧ трансмітер;
- 1 ІЧ приймач;
- 4 цифрових канали вводу / виводу;
- Ethernet порт 10 / 100;
- AxLinx гніздо.

Система:

- ультра-швидкий 304 MIPS процесор;
- 64 MB RAM;

- 32 Мб флеш-пам'яті;
- 512 КВ енергонезалежної пам'яті;
- розміри (4,01 см x 14,10 см x 13,00 см);
- вага 0,590 кг;
- живлення 280мА 12В постійного струму.

Отже, із важливих для нас характеристик:

- припустимі розміри;
- доступ до мережі Інтернет та підтримку протоколів FTP, SSH, HTTP, HTTPS/SSL та інші;
- 2 порти RS-485;
- достатні об’єми пам’яті;
- потужний мікропроцесор;
- багатопоточність;
- можливість перепрограмування віддалено;
- відома компанія, що має славу добре захищених від взлому та завадостійких пристроїв;
- контролер, що не має в собі рухомих складових, предназначений на довготривалу роботу без перезапусків в температурних умовах до 50 °С декілька десятків років;

Також, дуже важливим є те, що програмування цього контролеру та виробництвом інтерфейсів можлива за допомогою продукції виготовленої тією ж самою компанією. Отже, основною мовою програмування даних контролерів є NetLinx, а програмою для створення мобільних додатків або ж панелей управління, тобто графічних інтерфейсів – TPDesign4.

NetLinx – це назва власної мови програмування. Вона використовується при програмуванні пристроїв виготовлених компанією AMX.

NetLinx представляє собою розширену версію мови програмування Axcess. В ній додались нові типи даних, більше обробників подій, підтримка структур, багатовимірні масиви та інші функції.

Система керування NetLinx була розроблена для оновлення процесорної шини і підвищення потужності програмування мови Axcess. Спочатку вона була розроблена як нова версія та названа Axcess 2. Зв'язок між цими мовами досить подібний на зв'язок між C та C++. Так само, як C++ вніс новий рівень потужності для програмування на C, NetLinx пропонує безліч нових інструментів, команд, динамічно збільшує швидкість та потужність нинішніх та майбутніх додатків.

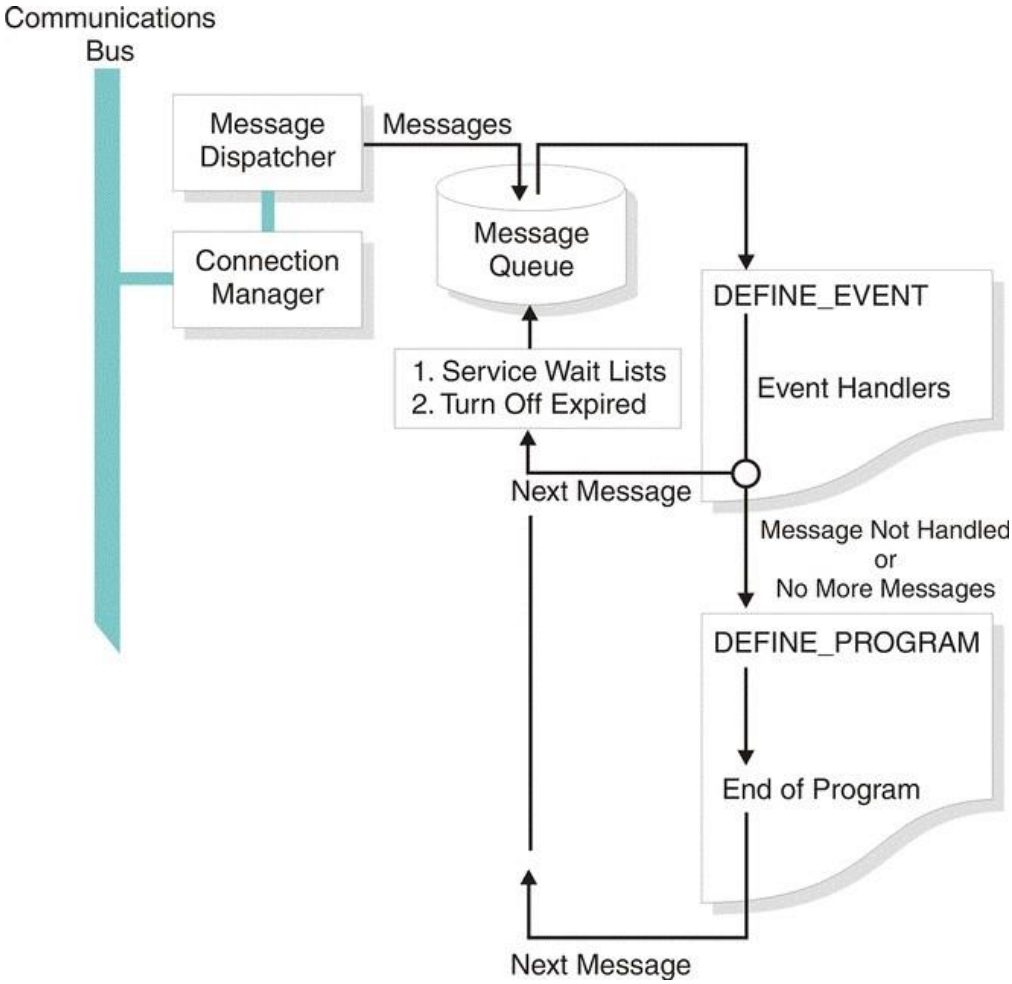


Рис. 2.5. Обробка повідомлень і основних процесів в системі NetLinx

Як бачимо з рисунку, то програмування на мові NetLinx дещо відрізняється від звичних для нас парадигм програмування, чи то ООП, чи то процедурне програмування, чи то системне. В основному, дана мова програмування побудована на реакції на певні повідомлення чи події. Тобто, спочатку виконуються стартові алгоритми, що записані в секції, що називається DEFINE_START.

Потім циклічно виконується частина коду, що записана в секцію DEFINE_PROGRAM. Але найцікавішими для нас являються ті функції, що записані в DEFINE_EVENT. Вони виконуються, в залежності від обраного нами варіанту, тоді, коли користувач натискає та кнопку, пристрій отримує команду або повідомлення, датчик засікає рух або ж термостат отримав дані про зміни температур.

Тобто, це складна система, що виконується без перестану та обробляє одночасно багато подій, та реагує на них.

Також, для створення графічного інтерфейсу використовується програма TPDesign.

Програма дизайну для сенсорної панелі TPDesign4 (або "TPD4") розроблена для того, щоб допомогти вам у створенні інтерфейсу сенсорної панелі або ж смартфону чи планшету для керування контролером.

Використовуйте TPD4 для створення файлів проекту (* .TP4), які містять всю інформацію, необхідну для визначення інтерфейсу користувача. Ця програма дає можливість створювати сторінки, спливаючі вікна та створеня логіки (навігації), а також кнопок і всіх файлів, пов'язаних з зображенням і звуком які використовуються при проектуванні (включаючи динамічні зображення та відео-канали).

Отже, з вибором контролеру ми визначились та оглянули середу програмування для праці з ним.

В нашому модулі контролер виконує функцію керування всіма приладами. Він буде відповідати за подачу електроенергії на акумулятор електроавтомобіля. Це буде виконано наступним чином:

- контролер буде отримувати дані з приладів збору інформації(лічильника електроенергії), сервера або ж з мобільного додатку користувача;
- зібравши всі дані контролер оброблятиме їх та буде розуміти, які дії потрібно виконувати і в якому випадку;
- дізнавшись про те, чи зараз потрібно подати електроенергію чи навпаки вимкнути її, контролер буде подавати на контактер сигнал замикання або ж розмикання відповідно.

Перш ніж обирати наступні прилади нашої електромережі потрібно проаналізувати ринок електромобілів та звернути увагу на те, як реалізовані певні зарядні пристрої.

Проблема полягає в тому, що в різних країнах світу використовується різні типи розеток та електромережі. Тому, різні компанії випускають різні штекери для зарядних пристроїв електромобіля.

Роз’єми зарядок залежать від способу підключення. Найбільш часто використовуваний роз'єм Mennekes призначений для підключення до трифазної розетки за допомогою кабелю.

Характеристика підключення:

- з 1-фазним кабелем: тип з'єднання SAE J1772;
- з 3-фазним кабелем: тип з'єднання Mennekes.

Отже ми вирішили надати можливість підключення зарядного приладу різних типів, щоб збільшити універсальність зарядного модуля.

Отож наступним елементом нашої системи являється реле. Електромагнітне реле є електричним ключем, керованим за допомогою електромагніту. При протіканні електричного струму через котушку реле перемикаються один або кілька ізольованих від котушки електричних контактів, комутуючи навантаження реле. Саме тому електромагнітне реле можна вважати універсальним комутатором аналогових та імпульсних сигналів.

Електромагнітне реле виконує наступні функції:

- гальванічна розв'язка між ланцюгом управління реле і ланцюгом навантаження реле;
- розмноження одного керуючого сигналу на кілька вихідних сигналів;
- посилення потужності сигналу, що управляє;
- незалежне управління декількома вихідними ланцюгами з різними рівнями струму і напруги (різними потужностями);
- розділення кіл з різними рівнями робочих струмів і напруг, а також ланцюгів змінного і постійного струму;
- перетворення і нормування рівнів електричних сигналів.

Але в нашій роботі потрібно використати потужне електромагнітне реле, що володіє важливою для промислової електротехніки здатністю комутації потужних електричних ланцюгів за допомогою малопотужного сигналу управління.

В якості приладу можна обрати будь – яке відоме вітчизняне реле. Ми будемо використовувати модуль силових реле REL8 компанії AVP-electro.

Пристрій має індикатор стану, кнопку перезавантаження і Dip-перемикач налаштування режимів роботи. На нижній платі знаходяться клеми для підключення харчування модуля, шини RS485, каналів входів і виходів.

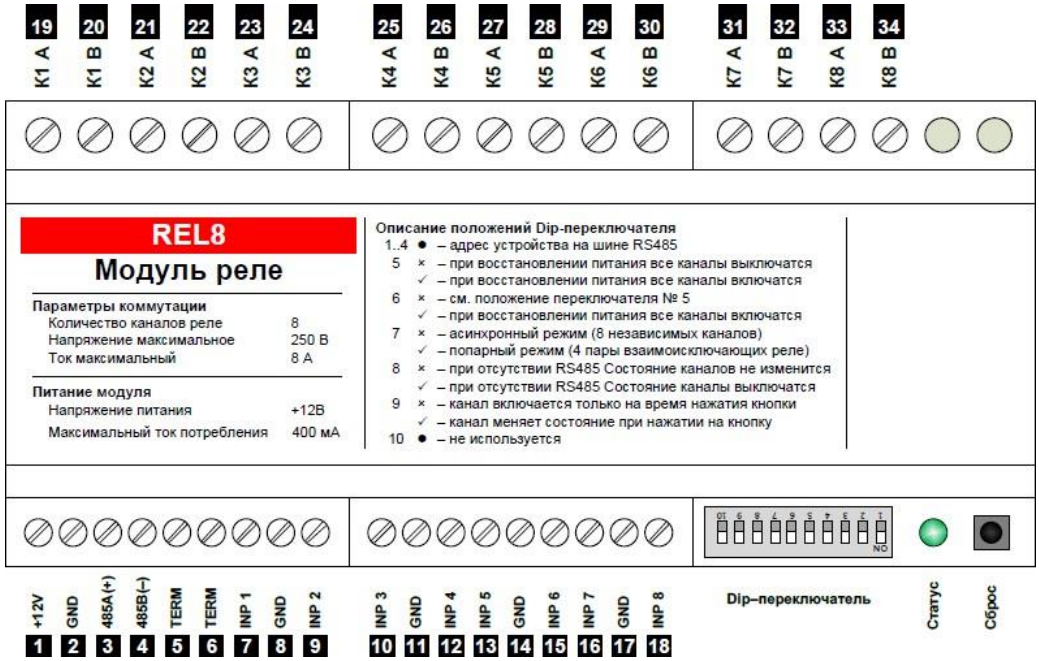


Рис. 2.6. Схематичне зображення модуля реле REL8

Як ми бачимо, даний пристрій має 8 незалежних каналів реле. Канали реле нормально-відкриті, тобто при відсутності живлення пристрої реле розімкнуті. Канали мають незалежний і попарний режим роботи. Цифрові входи нормально розімкнуті і можуть функціонувати в режимі «кнопка» і «вимикач».

В нашому випадку також доцільно буде використовувати не лише звичайні реле, а ще й їх підвид – контактори. Оскільки реле – низьковольтні електромагнітні комбіновані прилади.

Контактор - електромагнітний, електричний апарат, призначений для частих включень і виключень (до 3600 перемикань в годину) електричних силових ланцюгів постійного і змінного струму. Широко застосовується для дистанційного керування електричними машинами і апаратами в установках постійного і змінного струму при напрузі до 690В і силі струму до 1600А.

Отож, ми вже визначились з тим, що ми повинні надати можливість зарядки як однофазовим типам зарядного пристрою, так і трифазовим. Тому,

для цього ми повинні використовувати окремо два види контакторів. Один розрахований на однофазову передачу електроенергії. Інший - для трифазової.

Першим приладом будемо використовувати контактор – силове реле «Hager ESC125». Він розрахований на 220V, тому до цього контактора ми будемо підключати розетки європейського типу – розраховані на однофазову передачу електроенергії.

Рис. 2.7. Контактор – силове реле «Hager ESC125»

Основні характеристики, що цікавлять нас:

- 250V AC;
- 25A;

Іншим приладом будемо використовувати контактор – силове реле «Hager ESC340». До цього контактора ми будемо підключати розетки американського типу, тобто розетки, що розраховані на трифазову передачу електроенергії.

Рис. 2.8. Контактор – силове реле «Hager ESC300»

Основні характеристики, що цікавлять нас:

- 440V AC;
- 40A;

Також, для нашої системи потрібно обрати два різних види лічильника для підрахунку передачі кількості електроенергії на акумулятор. Тобто, знову ж таки, окремо лічильник для однофазової та окремо для трифазової подачі.

Основними характеристиками будуть:

- компактність;
- підключення по RS-485;
- відповідно однофазовий / трифазовий;
- електронний;

Для однофазової подачі ми знайшли досить гарний варіант. Він підходить по всім нашим параметрам та його компактність грає нам на руку. Цим варіантом був варіант лічильник електроенергії однофазовий з RS-485

LE-01MR. LE-01MR - це статичний (електронний) калібрований лічильник електроенергії з однофазним змінним струмом в прямій системі. Він використовується для індикації та реєстрації зібраної електрики і параметрів мережі електроживлення з можливістю дистанційного зчитування показань через дротову мережу стандарту RS-485.

Рис. 2.9. Лічильник електроенергії однофазовий LE-01MR.

Трьохфазові лічильники виявились більшими за габаритами та на порядок вищими в ціновому діапазоні. Тому, довелось вибирати з того, що маємо. Максимально підійшов по всім нашим вимогам наступний лічильник. Лічильник електроенергії трифазний LE-03M с MODBUS RTU, RS-485.

Лічильник серії М призначений для індикації та реєстрації значень спожитої електроенергії з можливістю дистанційного зчитування даних з лічильників по протоколу MODBUS RTU через послідовний інтерфейс RS485.

Рис. 2.10. Лічильник електроенергії однофазовий LE-03M.

Також, нам потрібно було обрати маршрутизатор для виведення нашого контролера в мережу та, можливо, для роздачі Wi-Fi сигналу для клієнта.

Звичайно в основі вибору роутера лежить співвідношення ціни до якості та мінімальних розмірів. І нашим виборов став маршрутизатор Mikrotik hAP mini.

MikroTik hAP mini – це новий бестселер від латвійської компанії MikroTik. Сам по собі hAP mini є логічним продовженням глобальної стратегії компанії з освоєння всіх сегментів ринку, вони почали захоплювати будинки більш старшими моделями і ось добралися до low price і обганяють за продажами вже звичні для ринку конкуруючі моделі від TP-Link і D- Link.

На ринку України MikroTik вже встиг завоювати лідируючі позиції і довів свою відданість користувачеві. Він впевнено працює вдень і вночі, в шторм і штиль і ніколи не залишить свого "юзера" без доступу в мережу.

MikroTik hAP mini - це недорогий Wi-Fi роутер на 2,4 ГГц для дому або невеликого офісу. Пристрій оснащений вбудованою все направленою антеною і не має строгого обмеження в кількості підключених клієнтів.

Важливим аспектом цього маршрутизатору для нас являються, адже модель Mini позначає не урізаний функціонал, а його дійсно компактність. Висота hAP mini становить всього 8 см.

У MikroTik hAP mini три мережевих порти, які налаштовані як один інтернет порт і два порти LAN, але, за бажанням, можуть бути налаштовані, використовуючи потужний конфігураційний функціонал RouterOS.

Живлення подається на маршрутизатор через порт microUSB за допомогою блоку живлення

В hAP mini встановлена операційна система RouterO, що володіє великими можливостями з налаштування. З її допомогою можна обмежити швидкість абонентам, заборонити доступ до необхідних сайтам або соцмереж, налаштувати VPN підключення, підключити другого провайдера для резервної мережі і багато іншого.

Якщо ви шукаєте недорогий і надійний Wi-Fi роутер для невеликого приміщення, то це досить гарний варіант.

Рис. 2.11. Маршрутизатор MikroTik hAP mini.

Також, нам звичайно знадобляться декілька силових кабелів, Ethernet-кабелі та RS-485.

RS-485 – протокол передачі даних. У стандарті RS-485 для передачі і прийому даних використовується одна кручена пара проводів, іноді супроводжується екранованою обшивкою або загальним проводом.

Передача даних здійснюється за допомогою диференціальних сигналів. Різниця напруги між провідниками однієї полярності означає логічну одиницю, різниця іншої полярності - нуль.

Стандарт RS-485 обумовлює тільки електричні і тимчасові характеристики інтерфейсу.

Стандарт RS-485 не обумовлює:

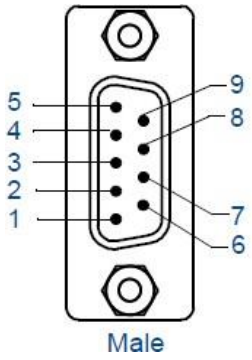
- параметри якості сигналу (допустимий рівень спотворень, відображення в довгих лініях),
- типи з'єднувачів і кабелів,
- гальванічну розв'язку лінії зв'язку,
- протокол обміну.

Ethernet – кабель, або ж звита пара – вид кабелю зв'язку. Являє собою одну або кілька пар ізольованих провідників, скручених між собою (з невеликим числом витків на одиницю довжини), покритих пластиковою оболонкою.

Звивання провідників проводиться з метою підвищення ступеня зв'язку між собою провідників однієї пари (електромагнітні перешкоди однаково впливають на обидва дроти пари) і подальшого зменшення електромагнітних перешкод від зовнішніх джерел, а також взаємних наведень при передачі диференціальних сигналів. Для зниження зв'язку окремих пар кабелю (періодичного зближення провідників різних пар) в кабелях UTP категорії 5 і вище дроти пари звиваються з різним кроком. Вита пара - один з компонентів сучасних структурованих кабельних систем. Використовується в телекомунікаціях і в комп'ютерних мережах як фізичне середовище передачі

сигналу в багатьох технологіях, таких як Ethernet, Arcnet, Token ring, USB. В даний час, завдяки своїй дешевизні і легкості монтажу, є найпоширенішим рішенням для побудови дротових (кабельних) локальних мереж.

Кабель підключається до мережевих пристроїв за допомогою роз'єму 8P8C (який помилково називають RJ45).



DB9 Serial Port pinouts (male connector)

RS-232	RS-422	RS-485
Pin 2: RX signal	Pin 1: RX -	Pin 1: A (strap to 9)
Pin 3: TX signal	Pin 4: TX +	Pin 4: B (strap to 6)
Pin 5: GND	Pin 5: GND	Pin 5: GND
Pin 7: RTS	Pin 6: RX +	Pin 6: B (strap to 4)
Pin 8: CTS	Pin 9: TX -	Pin 9: A (strap to 1)

Рис. 2.12. Схема підключення протоколів RS 232 / 422 / 285.

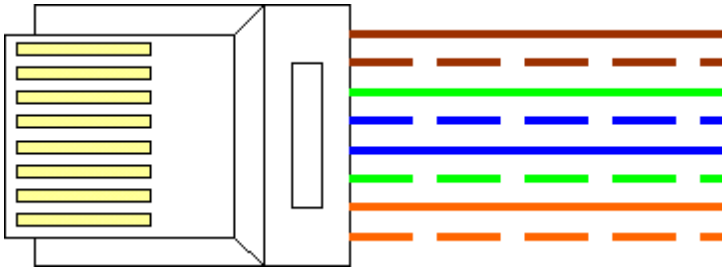


Рис. 2.13. Схема обжимання UTP кабеля.

2.3. Підключення всіх приладів у одну систему.

Спочатку потрібно налаштувати наший маршрутизатор, щоб забезпечити доступ до мережі нашому контролеру. Отже проводимо мережу до нашого роутера та виводимо по UTP кабелю наш контролер в Інтернет.

Тепер потрібно налаштувати наш контролер та підключити до нього інші прилади.

Основним протоколом спілкування всіх інших приладів з контролером являється RS-485. Отже підключаємо REL8 до контролера саме цим протоколом.

Контактори під’єднуємо на однофазну та трифазну електромережу відповідно до моделей. Також на виходи контакторів підводимо живлення.

Від контакторів підводимо лінії до лічильник, залежно від моделі та типу передачі струму та проводимо силові лінії до самих зарядних пристроїв (вилок) різного типу.

Схема системи буде наведена в додатку №1.

2.4. Огляд протоколу Modbus-RTU.

В нашій системі важливу функцію відіграє лічильник. Він підраховує кількість переданої електроенергії на акумулятор електромобіля. Цю інформацію він в той самий час передає на контролер. Оскільки обраний лічильник спілкується з контролером за допомогою протоколу Modbus-RTU, то потрібно оглянути його та розібрати на практичному прикладі.

Протокол ModBus, незважаючи на безліч спеціалізованих протоколів, що з'явилися в останні десятиліття, все ще займає лідируючі позиції в задачах автоматизації та диспетчеризації будівель і технологічних процесів. Багато фахівців вже звикли до його використання, незважаючи на його явну архаїчність. Проте, судячи зі статистики пошукових запитів і популярності статей про ModBus, явно існує пласт фахівців.

Перш за все, давайте розглянемо, як подаються дані в пристроях, які підтримують даний протокол. В наявності є чотири таблиці з даними:

Таблиця	Тип елемента	Тип доступу
Дискретні входи (Discrete Inputs)	один біт	тільки читання
Регістри прапорів (Coils)	один біт	читання і запис
Регістри введення (Input Registers)	16-бітове слово	тільки читання
Регістри зберігання (Holding Registers)	16-бітове слово	читання і запис

Рис. 2.14. таблиця даних Modbus-RTU.

У реальній практиці найчастіше зустрічаються пристрої, в яких є тільки таблиця Holding Registers, іноді об'єднана з таблицею Input Registers. Для доступу до цих таблиць існує ряд стандартний функцій ModBus.

Читання:

- 1 (0x01) - читання значень з декількох регістрів прапорів (Read Coil Status).
- 2 (0x02) - читання значень з декількох дискретних входів (Read Discrete Inputs).
- 3 (0x03) - читання значень з декількох регістрів зберігання (Read Holding Registers).
- 4 (0x04) - читання значень з декількох регістрів введення (Read Input Registers).

Запис одного значення:

- 5 (0x05) - запис значення одного прапора (Force Single Coil).
- 6 (0x06) - запис значення в один регістр зберігання (Preset Single Register).

Запис декількох значень:

- 15 (0x0F) - запис значень в кілька регістрів прапорів (Force Multiple Coils)
- 16 (0x10) - запис значень в кілька регістрів зберігання (Preset Multiple Registers).

Зі сказаного вище випливає, що найчастіше використовувані функції ModBus це 3, 6 і 16 («Read Holding Registers», «Preset Single Register» і «Preset Multiple Registers» - відповідно).

Розглянемо протокол ModBus RTU. Він застосовується для передачі даних по послідовним інтерфейсах, таким як RS-232 або RS-485. Більшість сучасних пристроїв використовують RS-485, так як він, по-перше, як правило,

двох провідний і по-друге, дозволяє підключити декілька пристроїв в один шлейф. Але в даному випадку використовується RS-232.

Важливим є те, що при подібній топології на одному шлейфі може бути тільки один ModBus Master, тобто пристрої не можуть вільно «спілкуватися» між собою. На кожному шлейфі організовується чітка ієрархія Master - Slave («Ведучий» - «Ведений»). Ведених, як уже було сказано, може бути кілька, а ведучий тільки один!

Адресна модель ModBus дозволяє використовувати адреси пристроїв з 1 по 247, що іноді вводить в оману деяких «проектологов», тому що RS-485 дозволяє підключити до однієї шини, без підсилювачів і репітерів, тільки 32 пристрої. Насправді, рекомендується не перевищувати значення 20 пристроїв на одну шину RS-232.

Отже, для читання одного Holding Register провідний надсилає запит на адресу відомого пристрою з кодом функції 3 (Read Holding Registers), зазначенням адреси цікавить регістра і кількістю регістрів для читання, в даному випадку = 1. На що ведений відповідає пакетом, в якому повторює власний адреса, номер оброблюваної функції і, в поле даних розміщує значення запитуваної регістра. Для читання декількох послідовних регістрів в запиті провідний просто вказує адресу першого і їх кількість.

У загальному вигляді, роботу функції 3 (Read Holding Registers) протоколу ModBus можна змалювати таку картину:

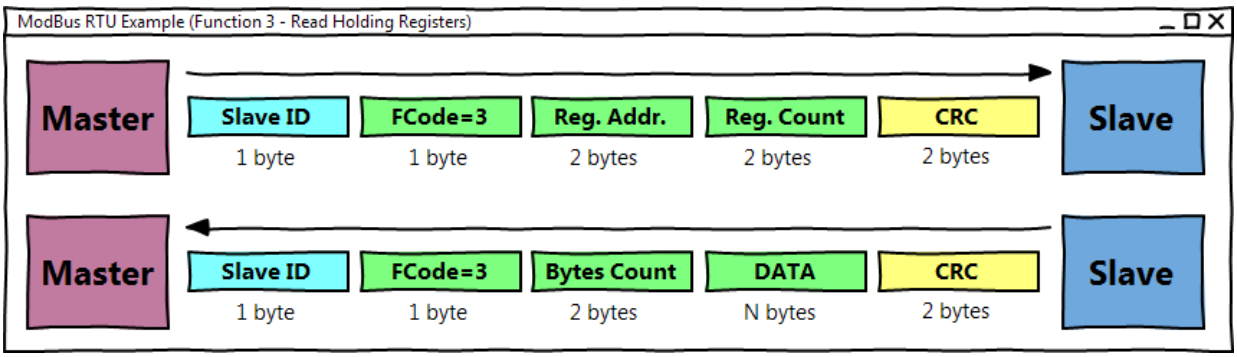


Рис. 2.15. приклад структури повідомлення Modbus-RTU.

2.5. Опис алгоритму програмування апаратної частини.

І так, ми маємо основну архітектуру системи та підключення всіх приладів між собою та до контролеру. Залишилось описати основний алгоритм спілкування пристроїв між собою та працю системи подачі та контролю електроенергією на зарядній станції.

Отже, наш алгоритм апаратної частини побудований на даних, що прийшли з сервера. Але роботу сервера ми розглянемо в наступному розділі нашої дипломної роботи, а зараз нас цікавить лише алгоритм апаратної частини.

Отже, контролер конкретної зарядної станції має дані про баланс на рахунку даного користувача системою. Ці дані він одразу переводить в еквівалент електроенергії, максимальну кількість якої може отримати електромобіль на даний момент.

На даному етапі може виникнути декілька варіантів розвитку подій.

Першим варіантом є те, що водій задав певну кількість електроенергії, яку бажає отримати. Одразу ж контролер перевіряє, чи має користувач на балансі стільки ж електроенергії. Якщо так, то починається процес зарядки електромобіля. В противному випадку виводиться попередження про те, що недостатньо коштів на балансі. Також користувачу буде запропоновано поповнити рахунок за допомогою спеціального сайту, а також зарядитися на всі кошти, які є на даний момент на рахунку юзера.

Іншим варіантом є вище названа зарядка автомобіля на всі доступні, на даний момент, кошти користувача.

Отже, на цьому етапі починається зарядка автомобіля. В обох випадках контролер має контролювати різницю вже наданої кількості електроенергії і кількість того, що він повинен передати. Щоб не виникло випадків, що електромобіль не зарядився на ту кількість, яку хотів отримати користувач, або ж, щоб за меншу ціну не продали надлишкову кількість електрики, ніж

було зазначено, щоб не було випадків, коли власник електрозарядного приладу йшов в мінус при продажі електроенергії, а також, щоб акумулятори користувачів не було пошкоджено.

Взагалі, як вже розглядалось раніше, в мові програмування NetLinx використовується система отримання інформації на DATA_EVENT або ж подібних подій. Тобто, коли виникає якась зміна інформації на якихось приладах (зміна температури на датчику, натискання кнопки на моніторі), то контролер отримує цю інформацію від приладів. Тоді, в залежності від того, який тип події відбувся виконується код певного івента.

Отже, в нашому випадку є декілька варіантів завершення процесу зарядки електромобіля, а саме:

- станція вже передала ту кількість електроенергії, яку було обрано користувачем;
- акумулятор вже повністю зарядженно;
- закінчення балансу користувача (актуально для зарядки на всі кошти, що має юзер на своєму віртуальному балансі);
- користувач припинив процес зарядки з мобільного додатку.

В залежності від типу ситуації виконуються різні події.

В першому випадку ми отримуємо інформацію по наданій вже на даний момент кількості електроенергії на електромобіль від нашого лічильника. Під час того, як відбувається подія DATA_EVENT(dvMeter), де dvMeter – це пристрій лічильника електроенергії, головний контролер отримує команду типу Modbus-RTU, що буде давати нам дані, про актуальні значення переданої електроенергії.

Виглядає це наступним чином:

```
DATA_EVENT(dvMeter)
{
   COMMAND:
   {
```

```
        modbusToKilowattHour(data.text);
        checkQuatityOfEnergy();
    }
}
```

Розглянемо деякі змінні та функції, а саме:

- data.text – це передана з лічильника команда, що сформована за правилами протоколу Modbus-RTU;
- modbusToKilowattHour() – це спеціальна функція, що отримуватиме повідомлення Modbus-RTU, розшифровуватиме його, та отримуватиме значення вже переданої кількості електроенергії. Також, це значення буде записано в певну глобальну змінну;
- checkQuatityOfEnergy() – це функція, яка буде перевіряти, чи глобальна змінна, яка має значення переданої електроенергії зчитаної з лічильника не перевищує задану кількість електрики, що потрібно надіслати на акумулятор електромобіля.

Наступний варіант, це той, коли акумулятор вже зарядився повністю. Сучасні електромобілі запрограмовані на то, що коли батарея повністю заряджена, то процес заряджання припиняється. Ми ж можемо лиш додати до цього те, що коли на лічильнику через певний час не змінюється кількість наданої електроенергії, то ми виводитимемо на екран повідомлення про те, що акумулятор було зарядженно і припиняти процес зарядки.

Також нами передбачено процес зарядки на всю кількість коштів, що має користувач на балансі на даний момент. Так, як зручніше всього зберігати інформацію про баланс користувача в тих самих одиницях вимірювання, що й вимірює лічильник, для подальшого порівняння його в тому самому івенті. Тобто, ми зводимо варіант того, що процес заряджання відбуватиметься на всі кошти, до того ж варіанту, що й заряджання на певну кількість kWh, лиш цю

кількість електроенергії обиратиме не користувач, а вона буде формуватися на основі актуального балансу користувача.

Отже третій варіант в нас буде оброблятися так само, як і перший.

Лишився останній варіант – примусова зупинка процесу заряджання електромобіля користувачем із додатку.

Тобто, в даний момент, на відміну від інших випадків ми оброблятимемо не подію отримування команд з боку лічильника, а обробка натискання певних кнопок. Тобто, код буде виглядати наступним чином:

```
BUTTON_EVENT(dvTPs, btn_stop)
{
   PUSH:
   {
      stopCharging();
   }
}
```

Розглянемо даний код. Спочатку в нас виконується BUTTON_EVENT(), це така сама подія, що й DATA_EVENT(), але вона обробляє натискання певної кнопки на певному пристрої. В даному випадку ми маємо наступні змінні та функції:

- dvTPs – це пристрій екранів \ мобільних телефонів, на яких будуть відбуватися натискання;
- btn_stop – це кнопка, номер каналу якого відповідає за зупинку процесу зарядки;
- stopCharging() – це функція, як відправляє команду на відповідний канал реле, що його потрібно вимкнути, тобто припиняємо процес зарядки електромобіля, що підключений до певної розетки;

Отже, ми розглянули певну структуру алгоритму обробки подій на основі яких буде вмикатися \ вимикатися процес зарядки. Також можна ознайомитися з алгоритмом на блок-схемі в додатку №2.

3. ПРОЕКТУВАННЯ ВЕБ-САЙТУ

3.1. Вибір стеку технологій для веб-сайту.

Розробка веб-сайту поділяється умовно на розробку фронтенд та бекенд частини проекту. Давайте спочатку розберемося, що ж це означає.

Фронтенд – це інтерфейс для взаємодії користувача та бекенду. Тобто, простими словами фронтенд відповідає за те, що може зчитувати браузер, інформацію, що він може вивести на екран та запускати її. Основними інструментами для розробки фронтенд частини є такі інструменти, як HTML, CSS та JavaScript.

HTML каже браузеру про зміст сторінки умовно поділяючи її на певні блоки, типу «заголовок», «список», «шапка сайту» і тому подібне.

CSS відповідає за відображення елементів. Тобто, основною функцією цього інструменту є стилізація сайту.

JavaScript обробляє взаємодію з користувачем та говорить браузеру, що робити в тому чи іншому випадку. Ця мова програмування дозволила сайтам перейти на новий рівень, тобто, основним трендом створення нових сайтів є те, що інформація з’являється без перезавантаження сторінки.

Тепер перейдемо до бекенду. Бекенд відповідає за те, що працює на сервері. Тобто, всі дії виконуються не на комп’ютері користувача сайтом, а на сервері, що дозволяє виконувати більш складні операції та обчисленя.

Для бекенду ми можемо використовувати будь-які інструменти, доступні на нашому сервері (який, по суті, є просто комп'ютером, налаштованим для відповідей на повідомлення). Це означає, що ми можемо використовувати будь-яку універсальну мову програмування, такі як: Ruby, PHP, Python, Java, JavaScript / Node, bash.

Це також означає, що нам потрібно використовувати системи управління базами даних, такі як MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached та інші.

Також, для того, щоб побудувати веб-сайт, нам потрібно розуміти взаємодію між фронтендом та бекендом.

На сьогоднішній день є декілька архітектур, які в основному використовуються в побудові сайтів.

Першим варіантом є серверні застосунки. Тобто, у цьому випадку HTTP-запити відправляються на сервер, а в свою чергу сервер дає відповідь у вигляді побудованої HTML-сторінки.

Зазвичай при такій архітектурі між отриманням запиту та відповіді сервер шукає по запиту інформацію в базі даних та генерує її за шаблоном.

Коли сторінка була завантажена в браузер, то різні інструменти відповідають по своєму за відображення інформації на фронтенді. Тобто, HTML відповідав за відображення всього контенту, CSS – за стилізацію тих чи інших елементів, а JavaScript – відповідав за всякі взаємодії між користувачем та даними на сторінці.

Така архітектура є досить застарілою, та не дуже часто використовується в сучасній побудові веб-застосунків. Це зумовлено тим, що для того, щоб дізнатися якусь нову інформацію потрібно повністю перезавантажити сторінку, а в випадках, коли якийсь певний блок сторінки обновляється циклічно та дуже часто це є досить неефективно, тому з часом було розроблено ще одну архітектуру.

Зв'язок з використанням AJAX.

Інший тип архітектури використовує ще один інструмент – AJAX. Отже, завантажений в браузер JavaScript надсилає HTTP – запит зсередини сторінки і отримує XML або ж JSON відповідь.

Також, це означає, що на сервері є частина коду, яка відповідає на запити JSON- або XML- кодів. Спеціальні протоколи, що використовуються для цього – REST та SOAP.

Така архітектура дозволяє побудувати деякі різновидності веб-застосунків.

Клієнтський веб-сайт – це такий веб-сайт, коли за допомогою AJAX досягнуто того, що дані на сторінці обновляються без перезавантаження сторінки. Частіше всього це використовується в таких фреймворках, як Angular, Ember та інше. Такий фронтенд спілкується з бекендом через HTTP, використовуючи JSON- та XML-відповіді.

Універсальні або ж ізоморфні застосунки.

Деякі бібліотеки та фреймворки, такі як React, надають вам можливість виконувати застосунки як насервері так і нафронтенді. В такому випадку для спілкування фронтенда з бекендом використовується як AJAX, так і оброблявані на сервері HTML сторінки.

Отже, нам потрібно обрати стек технологій для побудови нашого веб-застосунку. Окрім знайомих нам HTML, CSS та JavaScript з AJAX потрібно обрати мову програмування для бекенду, систему контроля базами даних, та, можливо, ще якісь фреймворки.

Так, як у нашому випадку, на місці фронтенду в нас використовується вбудований для нашого контролеру сервіс побудови управління системами, то нам не потрібен HTML, CSS та JavaScript з Ajax.

В нашому випадку дизайн та фронтенд додатку буде розроблено за допомогою програми TPDesign4. Але на рівні спілкування з бекендом цей варіант майже нічим не відрізняється.

Отже, нам потрібно обрати стек для бекенду. На даний момент нам потрібно обрати базу даних та мову програмування.

3.2. Вибір бази даних.

Першим етапом побудови серверної частини буде полягати в тому, щоб обрати базу даних. Для того, щоб обрати її потрібно заздалегідь продумати структуру даних, що ми будемо зберігати в ній.

Отож, вибір пав на декілька популярних баз даних, таких як:

- MySQL / MariaDB
- PostgreSQL
- MongoDB
- Redis

Отже, перш ніж розглянути кожен приклад, давайте обдумаємо наступне питання. Чи зручно буде для нас використовувати якусь нову не досить відому технологію чи краще використати щось, що ми вивчали на протязі навчання в університеті? Звичайно ж ні, краще використати практичні та теоретичні навички які ми маємо це вбереже нас від помилок та технічних нюансів, невідомих для нас.

Варіант один MySQL.

Народна СУБД або «must have», є практично на будь-якому хостингу. Проста в установці, працює нормально без особливих налаштувань. При належному підході може гнучко налаштовуватися під ваші потреби. Але є і підводні камені, в деякі випадках вона буде тим самим вузьким горлечком і ваш проект буде гальмувати, як би ви не тюнінгували СУБД і структуру даних.

Плюси MySQL:

- дуже проста в написанні для новачків;
- не потребує зусиль в установці;
- ми вивчали її на 3 курсі, отже маємо гарну базу знань для використання її на практиці;

- таблична система запису даних.

Також має свої мінуси, такі як:

- чутливість до нестабільності сервера. Особливо це впливає при використанні XtraDB від Percona. Якщо неправильно завершити MySQL, можна настільки поламати таблиці і бази даних, що відновити можна буде тільки з повного бекапа, звичайно, якщо ви їх регулярно робите;
- зміна структури даних може бути досить трудомістким процесом, особливо при великому кількість зв'язків між даними в різних таблицях і навіть при простому додаванні полів;
- посередня продуктивність.

Наступним варіантом є PostgreSQL.

Свого роду мастодонт, дуже стара і грамотна СУБД. Вона майже як MySQL, тільки краще. Але треба вміти використовувати і налаштовувати. За думку багатьох, дуже стабільна СУБД, її практично неможливо зламати, порушить таблиці як в MySQL. І це може бути для вас вирішальним фактором при виборі.

Плюси:

- дуже надійна БД;
- дуже добре встроктуровані дані з невеличкою гнучкістю;
- за допомогою сторонніх бібліотек просто і зручно розширюватися в кластери і робити шардінг таблиць. І все це дійсно працює.

Мінуси:

- необхідність досвіду роботи з цією СУБД, щоб приготувати її добре. Інакше краще взяти MySQL або прочитати далі;
- система авторизації за замовчуванням може викликати труднощі при використанні або налаштування, далеко не всім подобається,

деякі навіть дуже досвідчені розробники до сих пір не до кінця розуміють як воно там працює.

Третім нашим кандидатом був MongoDB.

В народі досить багато сперечань на рахунок того використовувати SQL чи NoSQL. Але все ж в деяких випадках MongoDB справляються із завданням набагато краще, ніж MySQL або PostgreSQL.

Плюси БД:

- у вас немає чіткої, заздалегідь описаної структури даних або ви припускаєте, що склад даних може потім сильно зміниться, тоді ця БД допоможе вам в цьому питанні без значних зусиль;
- гарно справляється з величезними об’ємами даних;
- просто тренд;
- легко встановити і спробувати, працює нормально без особливих налаштувань. А якщо заглибитися, вивчити, то налаштувати можна дуже багато чого.

Мінуси також має:

- відсутність простих транзакцій. Принаймні в тому, класичному вигляді, якими вони є в MySQL / PostgreSQL. При додаванні безлічі даних, які залежать один від одного, можуть бути певні труднощі, які доведеться вирішувати самостійно на рівні коду. Ну звичайно ви можете і не зіткнуться з такими проблемами, але ...;
- зв'язність даних практично відсутня. Відразу спадає на думку JOIN з SQL - ось цього по суті немає. Хоча, якщо бути більш точним, то треба просто мислити іншими категоріями;
- потрібно перебудовувати мислення саме під NoSQL.

Останній варант був Redis.

Найчастіше цю СУБД використовують в якості кешируючого шару для роботи з даними з іншої, більш повільної СУБД. Рідко, але все ж може використовуватися в якості самостійно БД для даних. Ця БД працює в пам'яті, дуже швидка, вміє зберігати дані на диску.

Підійде, бо:

- обсяг даних невеликий і дуже проста схема, укладає в шаблон «ключ = значення»;
- проста реалізація реплікації Master – Slave;
- реалізовано Pub / Sub (черги);
- можна використовувати як кеш для більш повільної СУБД або просто хочете не замислюватися про швидкість роботи СУБД з оглядкою на її обсяг.

Мінуси, куди ж без них:

- обсяг даних не повинен перевищувати обсяг вільного ОЗУ на вашому сервері (насправді може, але тоді всі вони будуть йти в swap, сильно уповільнюватимуть роботу, в загальному не слід);
- на догоду продуктивності присутнє досить слабке збереження даних. Тобто цілком може статися таке, що дані додали, а після рестарту їх немає;
- немає транзакцій в класичному їх розумінні;
- до сих пір не вміє нормально кластер і шардінг. Нормальної реалізації досі немає.

Оглядаючи всі варіанти, що були наведені вище, та зважаючи на наш досвід в написанні баз даних на MySQL, очевидним варіантом для нас буде саме ця система управління БД. Також, оскільки наша база даних не є досить важкою, та має не так багато полів тому ця база даних є досить гарним випадком.

3.3. Вибір мови програмування.

В наш час є досить багато різних мова для створення Back-end'а. Ось список найпопулярніших на даний момент:

- Java;
- Node js;
- .NET;
- Php;
- Python;
- Ruby.

Java – досить потужний, на плаву не перший рік, а це означає, що вона є досить надійною. Але, порівнюючи з іншими сучасними аналогами, код буде відносно повільний.

JsvsScript зараз став досить популярною мовою для створення сайтів. Спочатку він використовувався лише для фронтенду, але з часом почали розробляти різні фреймворки, такі як Node.js для серверної частини. Мова є досить перспективною та популярною.

.NET – фреймворк для мови C#. Відомий своєю схожістю з Java, але є швидшим її аналогом. Але щоб писати гарний код для серверної частини на цій мові потрібно знати багато моментів фреймворку.

Php – мова, яка в основному використовується для роботи з різними системами управління вмістом сайту, такі як WordPress та інші.

Python це така мова, яка є досить читабельною, простою для новачків та зазвичай код на ній є не достатньо габаритним.

Ruby – трошки схожий на Python. Основні переваги в тому, що він є досить швидким.

Для нашого проекту ми використаємо Node.js.

3.4. Проектування бази даних

Наступним кроком ми спроектуємо нашу базу даних. Так, як наш сайт є не досить громіздким, нам не потрібно використовувати досить багато полів.

Перша частина залежних один від одних таблиць в нашій базі даних - це дані про користувача та стан балансу.

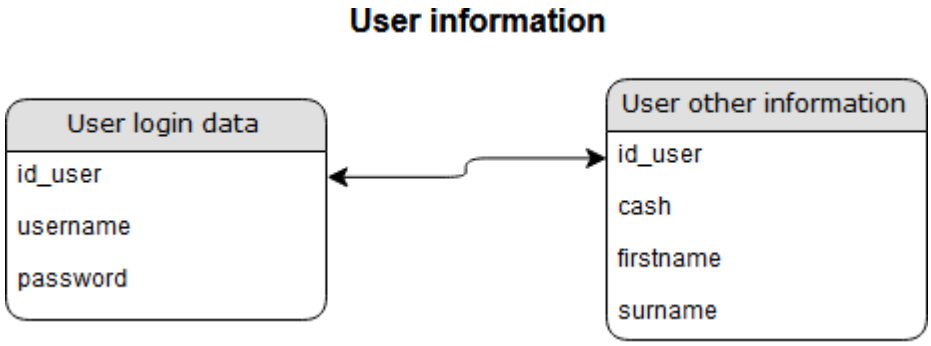


Рис. 3.1. схема залежності таблиць інформації про користувача.

Іншою частиною бази даних являються таблиці, що мають значення стану зарядних станцій та іншу інформацію про них.

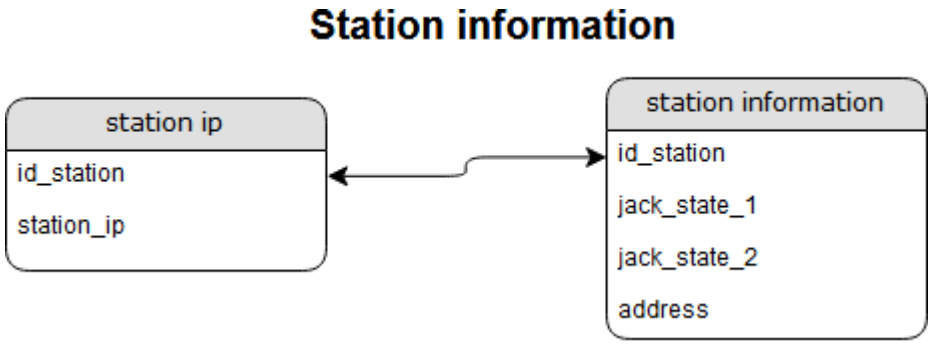


Рис. 3.2. схема залежності таблиць інформації про станції.

Суть наших залежностей в першому випадку являється досить простою. При авторизації юзера ми визначаємо id по якому знаходимо поле cash, в якому є значення балансу, на основі якого або проходить зарядка, або ж ні.

В наступному варіанті по ip контролера ми визначаємо id станції, що допоможе визначити стан конкретної розетки. Також, ми можемо дізнатися адресу даної зарядної станції.

3.5. Алгоритм спілкування контролера з сервером.

Отже, як працює наша система з боку серверного коду. Ми вже вияснили, що для того, щоб контролер знав, чи подавати електроенергію на контролер йому потрібно мати значення поточного стану на балансі користувача.

Для того, щоб контролер отримав ці значення, спочатку юзер повинен залогінитися в додатку. Контролер надсилає запит на отримання значення балансу на рахунку користувача. За допомогою значень полів username та password ми можемо знайти значення id_user за допомогою операторів SELECT, FROM, WHERE та ін.

Потім, знаючи id_user ми витягнемо значення поля cash із бази даних. У відповідь на запит контролера сервер надсилає значення балансу користувача, на основі яких контролер вже знає, подавати електроенергію на розетку чи ні.

ВИСНОВКИ

В ході виконання даного дипломного проекту було досліджено ринок електричних автомобілей, проаналізовано ряд компаній, що виготовляє свої електрозарядні станції та створено модуль зарядки для електромобілів, що врахував би всі плюси та мінуси існуючих аналогів

У першому розділі було проаналізовано загальний опис проблеми, готові рішення, які існують. Дослідження показали, що проблема з екологічними зарядними станціями існує та досить актуальна в наш час. Були розглянуті основні переваги та недоліки існуючих рішень даної проблеми. Але аналіз показав, що жоден з них не є достатньо самостійним та зручним варіантом для виведення на ринок.

Другий розділ дипломного проекту було присвячено особливостям розробки апаратної частини нашого модуля, а саме: наведені прилади, що ввійшли в систему, принцип їх роботи, проаналізовані основні переваги та недоліки. В даному розділі також було описано алгоритм роботи апаратної частини модуля.

В третьому розділі дипломного проекту було обрано стек технологій для створення серверної частини. Також в даному розділі було описано алгоритм спілкування контролера і сервера.

Поставлена задача була виконана успішно. Даний проект вже зараз може бути впроваджений на підприємство. Також варто зазначити, що архітектура даного рішення дозволяє легко кастомізувати функціонал під потреби підприємства.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Концепції розвитку інфраструктури електромобілів в Україні [Електронний ресурс]. – 2017. – Режим доступу: https://hevcars.com.ua/kontseptsiya-razvitiya-ryinka-elektrorazryadnyih-stantsiy-v-ukraine/ – Дата доступу: травень 2019.
2. Способи зарядки електромобіля [Електронний ресурс]. – 2016. – Режим доступу: https://ecotechnica.com.ua/stati/786-sposoby-zaryadki-elektromobilej-kak-eto-vse-rabotaet.html – Дата доступу: травень 2019.
3. STRUM [Електронний ресурс]. – 2015. – Режим доступу: https://strum.dtek.com/ – Дата доступу: травень 2019.
4. Дослідження компанії МОЭСК–EV [Електронний ресурс]. – 2018. – Режим доступу: https://www.dp.ru/a/2018/09/11/V_rasshirenii_seti_jelektro – Дата доступу: травень 2019.
5. Мікроконтролери [Електронний ресурс]. – 2018. – Режим доступу: http://electrik.info/main/automation/549-chto-takoe-mikrokontrollery-naznachenie-ustroystvo-princip-raboty-soft.html – Дата доступу: травень 2019.
6. Контроллер NI-700 [Електронний ресурс]. – 2017. – Режим доступу: https://www.cinos.net/wp-content/uploads/resources/amx/netlinx-controllers/cinos-amx-netlinx-ni-700-datasheet.pdf – Дата доступу: травень 2019.
7. Контактер [Електронний ресурс]. – 2017. – Режим доступу: http://www.svaltera.ua/magnitniy-puskatel/ – Дата доступу: травень 2019.
8. RS–485 [Електронний ресурс]. – 2016. – Режим доступу: https://www.compel.ru/lib/ne/2017/9/7-rs-485-vse-eshhe-samyiy-nadezhnyiy-promyishlennyiy-interfeys – Дата доступу: травень 2019.

9. Mikro Tik Routers [Електронний ресурс]. – 2017. – Режим доступу: https://mikrotik.com/products – Дата доступу: травень 2019.
10. Схема обжинки UTP кабеля [Електронний ресурс]. – 2017. – Режим доступу: http://bspsecurity.com.ua/text-3/kak-pravilno-obzhimat-kabel-vitaya-para/ – Дата доступу: травень 2019.
11. Вибір бази даних [Електронний ресурс]. – 2018. – Режим доступу: https://habr.com/ru/post/348220/ – Дата доступу: травень 2019.
12. Протокол Modbus-RTU [Електронний ресурс]. – 2018. – Режим доступу: https://ipc2u.ru/articles/prostye-resheniya/modbus-rtu/ – Дата доступу: травень 2019.
13. Протокол Modbus-RTU [Електронний ресурс]. – 2018. – Режим доступу: https://ipc2u.ru/articles/prostye-resheniya/modbus-rtu/ – Дата доступу: травень 2019.
14. Front and back ends [Електронний ресурс]. – 2018. – Режим доступу: https://en.wikipedia.org/wiki/Front_and_back_ends – Дата доступу: червень 2019.
15. Бази даних [Електронний ресурс]. – 2017. – Режим доступу: https://habr.com/ru/company/oleg-bunin/blog/358984/ – Дата доступу: червень 2019.
16. MySQL [Електронний ресурс]. – 2016. – Режим доступу: https://www.opennet.ru/docs/RUS/sql/#Syntax – Дата доступу: червень 2019.